A Study on Imputation-based Online Learning in Varying Feature Spaces

Jung-Hoon Lee
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea
jhlee0914@etri.re.kr

Cheol Ho Kim
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea
kimcheolho@etri.re.kr

Sungyup Lee
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea
sylee549@etri.re.kr

O.K. Baek
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea
ok.baek@etri.re.kr

***Abstract*—In this paper, we propose a new method for online learning in varying feature spaces (VFS) where the feature space of instances continually evolves. The proposed method, termed Online Imputation-based Learning (OIL), first imputes missing values and subsequently trains a classifier within a complete feature space. A significant advantage of this approach is the potential for considerable improvements by leveraging well-established research in both imputation and classification techniques. Furthermore, it offers the flexibility to easily modify model configurations based on specific conditions. The experimental results demonstrate that OIL not only performs comparably or even better than the state-of-the-art rival method in regularly varying data streams but also in arbitrarily varying data streams. This is evidenced across 13 benchmark datasets. Following this, we perform an ablation study under diverse conditions to further investigate the efficacy and robustness of various OIL configurations.**

***Index Terms*—online learning, varying feature spaces, imputation, ensemble learning**

I. Introduction

In recent years, there has been an exponential surge in the volume of data emanating from diverse sources. Leveraging traditional batch algorithms to learn and harness these data streams proves highly inefficient, primarily due to the necessity of periodic retraining to incorporate new generated data [1]. Beyond this computational inefficiency, storing the entire data streams for batch learning could precipitate memory constraints and data security issues, not to mention the myriad of logistical issues [2].

Given these challenges, the significance of online learning, which can incrementally learn data streams, is gaining considerable attention. Nonetheless, a prevalent shortcoming of traditional online learning methods is their assumption of a static feature space – an assumption often unrealistic in real-world scenarios. For instance, new data instances might introduce previously unseen features (new features), or they might lack features that previous instances had (missing features). Consider the context of a smart factory: sensors with innovative functionalities might be integrated, while existing ones may malfunction or reach their operational end. Furthermore, when countless users relay data to servers during IoT services, the data transmitted often varies subtly in configuration across users [3], [4]. This challenge is recognized as the varying feature spaces (VFS), and to fill this gap, several online learning algorithms have been proposed [3]–[7].

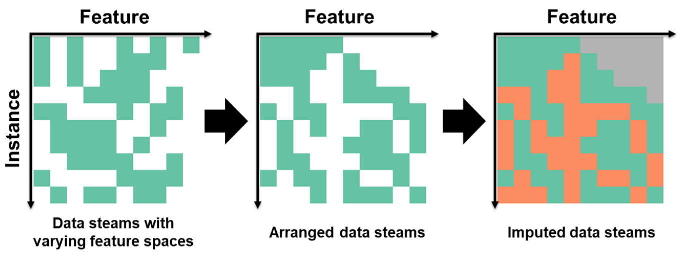

Fig. 1. Illustration of imputation (red zone) and padding (gray zone) process for data streams with varying feature spaces.

In this study, we explore a new method for learning in VFS, named Online Imputation-based Learning in varying feature spaces (OIL). Unlike previous methods that either completely exclude the imputation process [6], [7] or closely combine it with the classification process [4], [5], the main idea behind OIL is to conduct imputation and classification independently. Specifically, as shown in Fig. 1, OIL applies imputation to missing features (highlighted in the red zone) and to those features that have never been observed before, i.e., unobserved features (highlighted in the gray zone). This process transforms the data into a complete feature space without missing features, thereby enabling the deployment of a more expressive classifier. It's worth noting that in this paper, we reframe the VFS problem as a missing value issue, a concept well-established and familiar. However, it's surprising that this foundational perspective hasn't been extensively explored within the VFS context.

The rest of this paper is organized as follows: In Section II, we summarize related works on learning in VFS and imputation techniques. Section III provides the overall workflow of OIL, highlighting its differences from prior methods, and presents a detailed experimental methodologies. We then present the experimental results of OIL in comparison with the state-of-the-art (SOTA) method, and we will conduct relevant analyses in Section IV. The study concludes in Section V.

II. Related work

Given the significance of data streams with VFS, several online learning methods specifically designed for VFS have recently been proposed [3]–[7]. These methods can be broadly classified into three categories: passive-aggressive models (PAM), evolutionary ensemble models (EEM), and feature correlation models (FCM) [8]. Each has its unique advantages and disadvantages. Firstly, PAM [4] stands out for its solid mathematical foundation, often providing closed-form solutions. However, its performance becomes highly unstable with even a slight increase in missing features. As a result, FCM or EEM is usually employed for VFS with many missing features.

EEM [7] is an ensemble-based method that aggregates predictions from base models, each created based on individual features or subsets of features. One notable advantage of EEM is its robustness. EEM facilitates easy replacement of base models, enhancing its adaptiveness. However, there are associated limitations. EEM typically utilizes a few features for each base model. It is because, when larger subsets of features are utilized, the number of base models required in the ensemble can grow exponentially to accommodate diverse feature space variations. Consequently, only narrow feature correlation information can be learned.

Lastly, FCM [5] aims to reconstruct a complete feature space from a given feature subset. This is achieved by closely integrating imputation and classification processes. Specifically, the imputation model leverages both the reconstruction loss and the supervised loss propagated from the classifier for model updates. While FCM has impressive scalability, its application across diverse VFS scenarios predominantly relies on linear classifiers, which inherently presents performance constraints.

III. Experimental Methodology

The schema of OIL is shown in Fig. 2(a) and the other methods mentioned in Section II are illustrated together to highlight the differences between them. OIL combines the strengths of both FCM (Fig. 2(b)) and EEM (Fig. 2(c)). More specifically, by decoupling the imputation process from the classification process, OIL provides significant flexibility in classifier selection. Furthermore, OIL can effectively utilize feature correlation information as the classifier is trained on the complete feature space. In the following subsections, we will elaborate further on each step of OIL. The main process of OIL is implemented based on river [9] and scikit-learn [10] packages in python.

A. Varying feature space simulation

To validate the practicality of OIL, we selected 13 datasets that cover a wide range of properties: from small to large sizes,

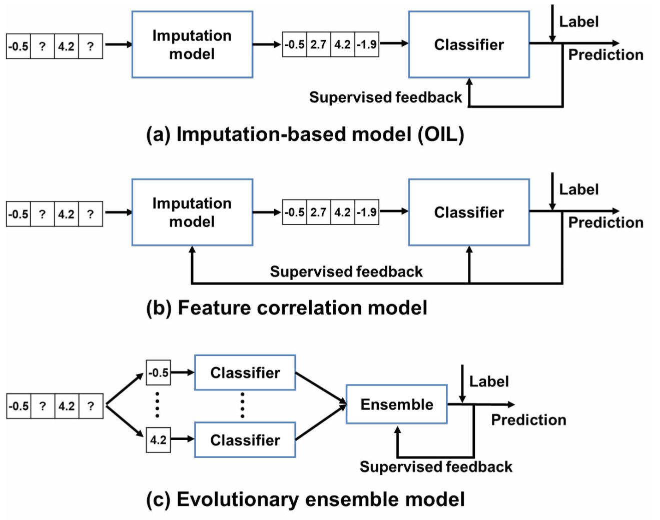

Fig. 2. Schema of OIL and the previous approaches to learning in varying feature spaces.

from low to high dimensions, from binary to multiclass, and across various domains. All datasets are available from the University of California, Irvine (UCI) ML Repository. The detailed information of the selected datasets is presented in Table I. OIL will be validated under two representative VFS

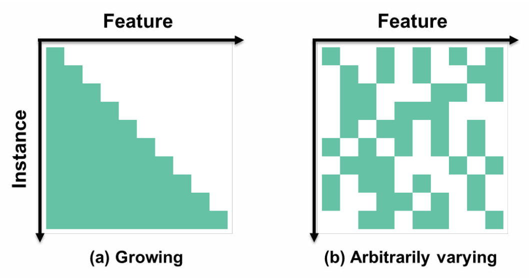

Fig. 3. Illustration of the two representative varying feature scenarios.

scenarios, Growing and Arbitrarily Varying, as shown in Fig. 3. Since the selected datasets have very few or no missing values, we simulate data streams with varying feature spaces by omitting some feature values from each instance. More specifically, in Growing, the whole dataset will be separated into 10 chunks, and the i-th chunk includes the first $i \times 10\%$ of features. For example, if a dataset has 1000 instances and 50 features, the first chunk contains the first 100 instances and the first 5 features. The 5th chunk contains the first 500 instances and the first 25 features. In arbitrarily varying scenario, we randomly omit a certain portion of features from each instance, resulting in random numbers of unobserved, overlapping, and new features. Unless stated otherwise, missing ratio r is set at 0.5. For a detailed performance trend analysis under different r, we will set r from 10% to 70% with a step of 20% in Section IV-C.

TABLE I
DESCRIPTION OF 13 DATASETS FOR EXPERIMENTS.

Name	Feature	Instance	Class
magic04	10	19020	2
svmguide3	21	1234	2
german	24	1000	2
wdbc	31	569	2
ionosphere	35	351	2
spambase	57	4601	2
wine	13	178	3
cardiotocography	21	2126	3
frogs	22	7195	4
robot24	24	5456	4
drybean	16	13611	7
optdigits	64	5620	10
texture	40	5550	11

B. Imputation process

In this section, we detail the selected imputation methods for our study and their adaptation to online learning. We selected five representative imputation methods based on a literature search, i.e., Mean, Mode, k-NN, MICE [11], MissForest [12]. The appropriate imputation method can vary greatly depending on the given tasks, datasets, and missingness patterns. Thus, we selected methods that are competitive under various conditions and can be adapted for online learning with minor modifications [13]–[15].

As a simple statistical imputation methods, mean and mode are popular due to their extremely low computational cost. Additionally, mean has the advantage that the online update results are exactly the same as batch update results.

For mode, when a feature has limited cardinality such as a categorical feature, mode can be updated by counting the occurrence of each value in online learning. However, for a continuous feature, since it might be difficult to know the number of unique values or the range of the feature in advance, so, we compute it through sliding window strategy which is often used for online imputation methods [16]. Unless otherwise specified, window size h is set to $min(500, 0.2n)$, where n is the total number of instances.

We selected k-NN as a representative machine learning method. k-NN impute missing values based on the k nearest neighbors from the learned instances. We set k to 5 and employ sliding window strategy, as previously described, to compose an instance pool for k-NN. Euclidean distance is employed to calculate pairwise distances, and any features with missing values are disregarded during the distance calculation.

Two iterative imputation methods are selected, MICE and MissForest, the main difference between them is that MICE perform multiple imputation and MissForest only perform single imputation. In our initial experiments, we found that these iterative methods are too slow for use in online learning, even when reducing the number of samples to learn or relaxing the convergence conditions. It might be because separate sub-models are needed for the imputation of each feature, and this process has to be repeated. Despite this drawback, the powerful performance of these two methods has led us to adopt a fixed window strategy, i.e., training the model only once using h initial instances for its practical application in online learning. Before store h instances for window, we employ k-NN for imputation.

Lastly, although deep learning-based imputation methods such as VAE or GAN are the subject of active research [15], we did not employ deep learning-based imputation methods in our study for several reasons. They typically require a large number of instances for training, which isn't practical in online learning [14]. Additionally, while gradient descent algorithms allow for updating deep learning models in online learning, such models are prone to catastrophic forgetting, resulting in unstable performance [17]. It's worth noting that we can effectively stabilize the imputation model using a shallow network and supervised loss, similar to GLSC [5]. However, exploring that approach is beyond the scope of our study.

C. Padding process

Unlike the typical imputation process in batch learning, the imputation process in VFS may results only in trapezoidal data streams. To further fill the data streams into a complete feature space, we need to pad the remaining parts that cannot be imputed due to the absence of previously observed feature values. In this study, for convenience, we refer to this additional imputation process exclusively required in VFS as the 'padding process' and evaluate three padding methods: Null, Zero, and MinMax.

Null padding method means that we opt not to pad the trapezoidal data streams. It's important to note that if we employ the Null method, the subsequent classifier must be capable of learning from trapezoidal data streams. We will pad all the remaining parts simply with zero in Zero method. Lastly, assuming we know the range of each feature in advance, we can employ MinMax method, which pads the remaining parts with either the 10% or 90% quantile values of each feature, selected randomly.

D. Classification process

Leveraging the flexibility of OIL, we've chosen adaptive random forest (ARF) [18], a widely recognized classifier for data streams, as our classifier in this paper. ARF is an ensemble method that utilizes very fast decision tree (VFDT) [19], the most popular Hoeffding tree algorithm, as its base model. It is well known that VFDT can handle new features for learning with minor modification [20]. Therefore, it also meets the requirements for the Null method mentioned in Section III-C.

The number of base models is set to 50 and each tree adopt local feature sampling to induce diversity. The maximum number of features for each node is set to $min(sqrt(m), m_t, 10)$, where m represents the number of features in the complete feature space and m_t is the number of observed features from the t-th instance. In other words, we limit the maximum number of features to 10 to maintain a reasonable computation speed even for high-dimensional data. Additionally, since we don't consider the concept drift that frequently occurs in non-stationary data, we don't employ any drift or warning detectors. The other hyper-parameters, such as instance resampling weight and node split threshold, are set to the default values of the package.

IV. Results

Experimental results and discussion are presented in this section. For online learning performance measurement, we use the cumulative error rate (CER) [7] as a metric and it is calculated based on the average of 5 trials with randomly shuffled datasets. Both feature and instance sequences are shuffled to reduce potential biases. Additionally, the evaluation results based on ORF^{3}V [7] are also included to provide a baseline. ORF^{3}V is a state-of-the-art online learner capable of handling both Growing and Arbitrarily Varying scenarios.

A. Comparison of different padding methods

As described in Section III-C, the padding process is always applied to trapezoidal data streams. Therefore, we directly study the effects of each padding method in Growing scenario. Since online imputation is not required in Growing, we can assess the impact of different padding methods more accurately than in Arbitrarily Varying. The results are shown in Table II.

TABLE II
Cumulative error rate based on each padding method in Growing scenario. The best results are highlighted in bold.

	ORF^{3}V	OIL-Null	OIL-Zero	OIL-MinMax
Name				
magic04	69.05	67.32	74.11	**74.27**
svmguide3	76.22	77.36	**78.16**	77.46
german	69.60	69.99	**71.13**	70.45
wdbc	86.68	86.83	**89.75**	87.43
ionosphere	73.51	**83.54**	83.49	78.63
spambase	69.32	83.46	**86.10**	84.47
wine	70.68	69.94	**72.20**	70.73
cardiotocography	77.78	80.26	**82.62**	80.47
frogs	75.74	**87.56**	87.48	87.23
robot24	65.25	**81.61**	75.99	70.50
drybean	52.53	81.73	81.74	**83.11**
optdigits	60.38	69.92	**73.92**	68.73
texture	46.75	**83.66**	79.38	78.51

We observe that the OIL method showed performance comparable to ORF^{3}V in all datasets. Among OIL methodologies, the performance based on the Zero padding method is on average the highest (highest in 7 out of 13 datasets). Additionally, OIL-Zero demonstrated higher performance than ORF^{3}V in all datasets. Of course, in datasets like ionosphere and frogs, the performance of OIL-Null is the highest, but it showed lower performance than ORF^{3}V in datasets like wine and magic04. It is because, in OIL-Null, unobserved features don't participate in training. This means that all trees tend to be trained on similar feature subsets, reducing the diversity, which is critical for ensemble methods. In contrast, filling in any value with Zero or MinMax padding methods allows each tree to have different feature subsets, making an effective ensemble possible. Furthermore, it can delay learning until there's enough data for a split operation, allowing for more cautious training.

Lastly, even though OIL-MinMax utilize additional information through quantile values, there is no significant performance improvement. This is because quantile values might act as noise against real data. These results imply that relying on conservative assumptions can produce overall better results than utilizing incomplete additional information.

Considering the overall findings regarding the padding methods, we will use the Zero padding method in subsequent experiments.

B. Comparison of different imputation methods

Based on experimental settings described in Section III, we evaluate OIL under Arbitrarily Varying scenario in this section. It's worth noting that each method doesn't necessarily need to reconstruct the original value accurately, as long as it can improve CER. The results are shown in Table III.

From our experimental results, we observed that using just OIL-Mean yields superior performance in 10 out of the 13 datasets when compared to ORF^{3}V. Both OIL-k-NN and OIL-MissForest surpass ORF^{3}V in 12 out of the 13 datasets. This suggests that even with basic imputation methods, one can achieve performance comparable to ORF^{3}V. It underscores the idea that while novel algorithms might have diverse advantages, leveraging existing research with the OIL approach can already deliver commendable CER in VFS scenarios.

It is interesting that there isn't a significantly superior OIL method. Among 13 datasets, OIL-Mean and OIL-MICE each showed the highest performance in one dataset, OIL-Mode in two, OIL-k-nn in four, and OIL-MissForest in six datasets. This demonstrates the importance of choosing imputation methods suited to the dataset at hand.

Nonetheless, unlike OIL-Mode, which shows significant performance variations depending on the dataset, OIL-Mean provides stable performance across datasets and is comparable to ORF^{3}V. Therefore, it would be an excellent choice in situations with extremely limited computational resources. If there are some computational resources available and it's possible to store hundreds of instances, then OIL-k-NN or OIL-MissForest are good choices. Especially k-NN, with numerous related online learning studies existing [21]–[23] and its high flexibility as a type of lazy learning, can be utilized in various situations. Lastly, while OIL-MissForest requires the relatively high computational effort, it consistently delivers the highest performance across datasets or, at the very least, provides performance that's comparable to the best. Thus, it can be a viable option depending on the situation.

C. Ablation study on missing ratio

To further investigate the usefulness of various imputation methods in VFS, we selected the Mean, k-NN, and MissForest methods and evaluated them under different missing ratios: 0.1, 0.3, 0.5, and 0.7, in the Arbitrarily Varying scenario. The results are presented in Table IV. While not presented in the main text, we also conducted experiments for the case where

TABLE III
CUMULATIVE ERROR RATE BASED ON EACH IMPUTATION METHOD IN ARBITRARILY VARYING SCENARIO. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Name	ORF^{3}V	OIL-Mean	OIL-Mode	OIL-k-NN	OIL-MICE	OIL-MissForest
magic04	70.25	75.74	74.42	74.64	75.90	**75.92**
svmguide3	76.22	**77.29**	77.10	76.81	76.99	77.26
german	**69.89**	69.75	**69.89**	69.75	69.85	**69.89**
wdbc	89.53	90.46	90.28	91.94	**92.43**	91.94
ionosphere	73.17	77.94	74.51	80.51	80.74	**80.80**
spambase	68.42	83.01	**86.15**	82.28	84.88	84.66
wine	74.39	71.98	59.89	**76.61**	75.59	76.16
cardiotocography	78.00	79.12	79.11	79.49	82.75	**82.87**
frogs	75.06	84.70	78.06	**91.30**	90.53	90.67
robot24	66.03	62.11	58.93	68.54	68.27	**69.01**
drybean	54.80	84.61	76.35	78.85	87.45	**87.53**
optdigits	64.84	68.38	54.62	**77.42**	73.87	74.71
texture	51.59	70.25	62.55	**76.40**	75.38	75.42

r=0.9. However, the vast amount of missing data appears to make meaningful learning challenging, regardless of the imputation method applied, so we excluded it from this paper.

TABLE IV
CUMULATIVE ERROR RATE IN ARBITRARILY VARYING SCENARIO WITH DIFFERENT MISSING RATIOS. THE PERCENTAGE RELATIVE TO THE MAXIMUM PERFORMANCE IS INDICATED INSIDE THE PARENTHESES.

Imputation	r	robot24	optdigits	texture
OIL-Mean	0.1	83.48 (1.00)	83.52 (1.00)	80.14 (1.00)
	0.3	74.17 (0.89)	77.52 (0.93)	75.83 (0.95)
	0.5	64.23 (0.77)	66.40 (0.80)	69.81 (0.87)
	0.7	56.06 (0.67)	48.16 (0.58)	56.23 (0.70)
OIL-k-NN	0.1	84.20 (1.00)	87.10 (1.00)	82.41 (1.00)
	0.3	77.20 (0.92)	85.18 (0.98)	81.43 (0.99)
	0.5	68.43 (0.81)	77.70 (0.89)	76.83 (0.93)
	0.7	59.93 (0.71)	55.95 (0.64)	62.34 (0.76)
OIL-MissForest	0.1	84.20 (1.00)	86.55 (1.00)	80.89 (1.00)
	0.3	76.72 (0.91)	83.15 (0.96)	79.61 (0.98)
	0.5	68.85 (0.82)	74.69 (0.86)	74.85 (0.93)
	0.7	58.74 (0.70)	57.22 (0.66)	66.36 (0.82)

Upon examining the results, we note that when the missing ratio $r = 0.1$, the performance differences among imputation methods are not significant. However, as r increases, these differences become more pronounced. Specifically, Comparing the scenario where $r = 0.7$ with that of $r = 0.1$, we observe that MissForest maintains its performance most effectively, showing an average performance drop of about 27%. In comparison, k-NN experienced a drop of 30%, and Mean dropped by 35%.

On the other hand, when r equals 0.3 or 0.5, the performance drop of k-NN is slightly less pronounced, at 4% and 12%, respectively. This suggests that while k-NN is effective when r is 0.3 or 0.5, MissForest, which leverages information from all available instances, becomes more beneficial as the number of missing values increases. It also explains why the Mean method, being the simplest and a kind of column-wise imputation technique, consistently demonstrates the poorest ability to maintain performance across all tested values of r. Consequently, as r increases, the robustness of the methods can be ranked as MissForest, followed by k-NN, and then the Mean method.

D. Ablation study on computational resource

In previous experiments, we have already demonstrated that various imputation methods, especially k-NN and MissForest, can achieve excellent performance. However, these two methods require the use of a sliding window strategy for online learning. Additionally, in the ensemble classifier part, the more base models we employ, the greater the computational resources needed—this is not ideal in an online learning setting. Therefore, in this section, we will explore the performance variations when relaxing the conditions on sliding window size and ensemble size. By comparing this OIL performance with the ORF^{3}V results obtained in Section IV-B, we can determine the minimal conditions required to achieve performance comparable to ORF^{3}V.

TABLE V
CUMULATIVE ERROR RATE IN ARBITRARILY VARYING SCENARIO ACROSS VARIOUS ENSEMBLE SIZES.

# of tree / Name	5	10	20	30	40	ORF^{3}V
robot24	60.90	64.64	67.60	67.22	67.71	66.03
optdigits	61.27	69.34	74.10	75.85	77.21	64.84
texture	74.09	75.37	75.73	76.53	76.10	51.59

When examining the impact of ensemble size on CER, as presented in Table V, we find that for robot24 dataset, employing a mere 20 base models is enough to achieve comparable performance to ORF^{3}V. The corresponding CER of ORF^{3}V is provided on the far right of Table V for comparison. Remarkably, optdigits and texture required only 10 and 5 base models, respectively.

To study the impact of window size, we varied h from $0.02n$ to $0.10n$ with a step of $0.02n$ for the k-NN imputation method, where n denotes the total number of instances. The results are shown in Table VI. Notably, For the robot24, optdigits, and texture datasets, a window size of $0.06n$, $0.02n$, and $0.02n$ respectively yields performance comparable to that of ORF^{3}V.

TABLE VI
CUMULATIVE ERROR RATE IN ARBITRARILY VARYING SCENARIO ACROSS VARIOUS WINDOW SIZE h.

h Name	$0.02n$	$0.04n$	$0.06n$	$0.08n$	$0.10n$	ORF^{3}V
robot24	61.62	64.09	66.05	67.95	68.56	66.03
optdigits	72.62	75.31	76.61	77.11	77.33	64.84
texture	71.95	74.32	75.53	75.75	76.50	51.59

V. CONCLUSION

In this study, we introduce a novel approach named OIL for learning in varying feature spaces. Through comprehensive experiments on 13 benchmark datasets, we demonstrated OIL's superiority over the SOTA method. Specifically, OIL, utilizing basic Mean imputation and Zero padding without any additional assumptions, outperforms ORF^{3}V in 10 out of the 13 datasets. Moreover, OIL, when implemented with MissForest imputation and Zero padding (which demands an assumption for slight data storage and computational resources), surpasses ORF^{3}V's performance in 12 out of 13 datasets. This suggests that OIL could be a viable solution for addressing classification in VFS.

Considering the practicality of online learning, we believe the window-based k-NN (which also outperforms ORF^{3}V in 12 out of the 13 datasets) is the most attractive option. As future work, we intend to enhance the k-NN by introducing prototype or clustering-based augmentation to make it even more suitable for online learning. From the classifier's perspective, while we have currently tuned the ARF minimally, we plan to further refine it based on the requirements of VFS. Additionally, we aim to improve OIL's overall performance by considering advanced ensemble base models, like moving from VFDT to subsequent versions such as EFDT.

ACKNOWLEDGMENT

This work was supported by Electronics and Telecommunications Research Institute(ETRI) grant funded by the Korean government (23ZS1100, Core Technology Research for Self-Improving Integrated Artificial Intelligence System).

REFERENCES

[1] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[2] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, 2020.

[3] S. Gu, Y. Qian, and C. Hou, "Learning with incremental instances and features," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[4] E. Beyazit, J. Alagurajah, and X. Wu, "Online learning from data streams with varying feature spaces," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3232–3239.

[5] Y. He, B. Wu, D. Wu, E. Beyazit, S. Chen, and X. Wu, "Toward mining capricious data streams: A generative approach," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 3, pp. 1228–1240, 2020.

[6] C. Schreckenberger, C. Bartelt, and H. Stuckenschmidt, "Dynamic forest for learning from data streams with varying feature spaces," in *International Conference on Cooperative Information Systems*. Springer, 2022, pp. 95–111.

[7] C. Schreckenberger, Y. He, S. Lüdtke, C. Bartelt, and H. Stuckenschmidt, "Online random feature forests for learning in varying feature spaces," vol. 37, no. 4, pp. 4587–4595.

[8] H. S. Yi He, Schreckenberger and X. Wu, "Towards Utilitarian Online Learning – A Review of Online Algorithms in Open Feature Space," in *Proc. of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[9] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse *et al.*, "River: machine learning for streaming data in python," vol. 22, no. 110, pp. 1–8.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel *et al.*, "Scikit-learn: Machine learning in python."

[11] S. V. Buuren and K. Groothuis-Oudshoorn, "**mice** : Multivariate imputation by chained equations in *R*," vol. 45, no. 3.

[12] D. J. Stekhoven and P. Bühlmann, "MissForest—non-parametric missing value imputation for mixed-type data," vol. 28, no. 1, pp. 112–118.

[13] S. Jäger, A. Allhorn, and F. Bießmann, "A benchmark for data imputation methods," vol. 4, p. 693674.

[14] Y. Sun, J. Li, Y. Xu, T. Zhang, and X. Wang, "Deep learning versus conventional methods for missing data imputation: A review and comparative study," vol. 227, p. 120201.

[15] A. Nazábal, P. M. Olmos, Z. Ghahramani, and I. Valera, "Handling incomplete heterogeneous data using VAEs," vol. 107, p. 107501.

[16] W. Dong, S. Gao, X. Yang, and H. Yu, "An exploration of online missing value imputation in non-stationary data stream," vol. 2, no. 2, p. 57.

[17] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," vol. 113, pp. 54–71.

[18] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger *et al.*, "Adaptive random forests for evolving data stream classification," vol. 106, no. 9, pp. 1469–1495.

[19] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 71–80.

[20] C. Schreckenberger, T. Glockner, H. Stuckenschmidt, and C. Bartelt, "Restructuring of hoeffding trees for trapezoidal data streams," in *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, pp. 416–423.

[21] S. Eghbali, H. Ashtiani, and L. Tahvildari, "Online nearest neighbor search in binary space," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 853–858.

[22] ——, "Online nearest neighbor search using hamming weight trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 7, pp. 1729–1740, 2019.

[23] P. Raja and K. Thangavel, "Missing value imputation using unsupervised machine learning techniques," *Soft Computing*, vol. 24, no. 6, pp. 4361–4392, 2020.